# OSSに対するジャンプ拡散過程モデルに基づく保守労力の最適化手法 (不確実性の下での意思決定の数理とその周辺)

AUTHOR(S):

田村, 慶信; 曽根, 寛喜; 山田, 茂

CITATION:

田村, 慶信 ...[et al]. OSSに対するジャンプ拡散過程モデルに基づく保守労力の最適化手法 (不確実性の下での意思決定の数理とその周辺). 数理解析研究所講究録 2019, 2126: 174-180

ISSUE DATE:

2019-08

URL:

http://hdl.handle.net/2433/252247

数理解析研究所講究録 第2126巻 2019年 174-180



# OSSに対するジャンプ拡散過程モデルに基づく保守労力の最適化手法

東京都市大学・知識工学部 田村 慶信（Yoshinobu Tamura）†
†Faculty of Knowledge Engineering, Tokyo City University
東京都市大学・総合理工学研究科 曽根 寛喜（Hironobu Sone）††
††Graduate School of Integrative Science and Engineering, Tokyo City University
鳥取大学・名誉教授 山田 茂（Shigeru Yamada）†††
†††Emeritus Professor, Tottori University

## 1 はじめに

低コストかつ短納期により開発できるオープンソースソフトウェア（Open Source Software，以下OSSを略す）を利用したソフトウェアサービス環境が多用されている．しかしながら，「安く早く作れる」というメリットだけが優先してしまい，そのセキュリティや品質上の問題に多くの企業が悩まされているという現状がある．こうしたOSSを利用したソフトウェアシステムの品質を定量的に評価する手法は未だ提案されておらず，職人的・試行錯誤的に運用が行われているのが現状である．また，近年のOSSの運用環境について考えた場合，OSS単体ではなく様々な連携ソフトウェアとのネットワーク経由での通信状況などを考慮することは非常に重要となる．本論文では，ビッグデータを扱うOSSの運用上の特徴を包括するためのジャンプ拡散過程モデルを提案する．

ビッグデータを考慮したOSSに対する最近の研究動向としては，サービス形態，性能評価等を対象とした文献はいくつか提案されている [1]．ビッグデータを有するOSS開発をプロジェクトマネジメントの観点から開発工数に基づき定量的に評価する手法は未だ提案されていない．特に，ビッグデータを考慮することを考えた場合，開発保守のための工数が大きくなるだけでなく，ログインするユーザの突発的な増減なども考慮する必要がある．ビッグデータを運用するOSSプロジェクトの場合，そこに潜む規則性や秩序を見い出し，複数のパラメータをもつ数理モデルを適用することは可能であっても，パラメータ推定などの問題から，実利用上適用することは非常に難しい．このような，ビッグデータを有するオープンンソースプロジェクト特有の状況に対して，定常的なノイズと突発的なノイズの2種類の雑音を適用する．これにより，ビッグデータからの外的要因を考慮しつつ，大規模OSSシステムを運用するプロジェクトを定量的に評価することが可能となる．

ソフトウェアの信頼性を評価するための数理モデルとして，これまでに数百におよぶソフトウェア信頼性モデルが提案されてきた [2,3]．ソフトウェア信頼度成長モデルはフォールトデータに基づいてソフトウェア信頼性が評価されるが，フォールト発生の原因となるプロジェクト状態を定量的に評価することができれば，ソフトウェア開発と運用においてQCD（Quality，Cost，Delivery）の観点から最適化を図ることが可能となるものと考える．

本論文では，開発と保守労力の観点から，提案されたジャンプ拡散過程モデルに基づく開発・保守労力の最適化問題を考える．OSSの運用に伴う開発・保守労力に基づく最適メンテナンス時刻を推定することにより，OSS開発と運用においてQCDの観点からソフトウェア品質の向上を図ることが可能となる．さらに，実際の開発工数データを利用した提案手法の数値例を示す．

## 2　ジャンプ拡散過程モデル

時刻 $t=0$ で OSS の運用が開始され，任意の時刻 $t(t\geq 0)$ までの投入開発工数 $\Lambda(t)$ は以下の常微分方程式によって記述されるものと仮定する．

$$\frac{d\Lambda(t)}{dt}=\beta(t)\{\alpha-\Lambda(t)\}. \tag{1}$$

ここで，$\beta(t)$ は時刻 $t$ における開発工数の変化率を，$\alpha$ は OSS の特定バージョンのリリースに必要とされる開発工数を表す．OSS プロジェクトでは，開発者の習熟度のばらつきやネットワーク環境の不安定な状態に影響され，実際の開発工程は不規則な性質を示すと考えられる．よって，リリースまでに必要とされる開発工数の変化率は開発期間を通じて必ずしも一定であるとは限らない．そこで，開発工程のもつそのような確率的性質を，開発工数の変化率 $\beta(t)$ の時間的な不規則変動でモデル化する．このとき，式 (1) は，

$$\frac{d\Lambda(t)}{dt}=\{\beta(t)+\sigma\nu(t)\}\{\alpha-\Lambda(t)\}, \tag{2}$$

となる [4]．$\sigma$ は定数パラメータである．$\nu(t)$ は解過程の Markov 性を保証するための標準化された Gauss 型白色雑音である．さらに，ログインするユーザの突発的な増減やマイナーバージョンアップのような急激な開発工数の変化などの影響を考慮し，ジャンプ項を導入する [5]．式 (2) を，以下の Itô 型の確率微分方程式 [4,6] に拡張して考える．

$$d\Lambda(t)=\left\{\beta(t)-\frac{1}{2}\sigma^2\right\}\{\alpha-\Lambda(t)\}dt+\sigma\{\alpha-\Lambda(t)\}d\omega(t)+d\left(\sum_{i=1}^{M_t(\tau)}(V_i-1)\right). \tag{3}$$

ここで，$M_t(\tau)$ は，$\omega(t)$ とは独立な強度パラメータ $\tau$ をもつポアソン過程であり，時刻 $t$ までにジャンプが発生した回数を表す．$\tau$ はジャンプ事象が生じる確率的な頻度である．また，$V_i$ は $i$ 回目のジャンプ幅を表す独立な確率変数である．式 (3) の確率微分方程式を Itô の公式を用いて変換すると，

$$\Lambda(t)=\alpha\left[1-\exp\left\{-\int_0^t\beta(s)ds-\sigma\omega(t)-\sum_{i=1}^{M_t(\tau)}\log V_i\right\}\right], \tag{4}$$

となる [7]．本論文では，開発工数の変化率 $\beta(t)$ は，簡単のために既存のソフトウェア信頼度成長モデルにおける平均値関数を流用することにより，次式を満たすものとする．

$$\begin{aligned}\beta(t) &\doteq \frac{\frac{dF_*(t)}{dt}}{\alpha-F_*(t)},\\ F_e(t) &= \alpha(1-e^{-\beta t}), \qquad (5)\\ F_s(t) &= \alpha\left\{1-(1+\beta t)e^{-\beta t}\right\}. \qquad (6)\end{aligned}$$

ここで，$\beta$ は開発工数の変化率を表す．

したがって，投入開発工数のサンプルパスは，

$$\begin{aligned}\Lambda_e(t) &= \alpha\left[1-\exp\left\{-\beta t-\sigma\omega(t)-\sum_{i=1}^{M_t(\tau)}\log V_i\right\}\right], \qquad (7)\\ \Lambda_s(t) &= \alpha\left[1-(1+\beta t)\exp\left\{-\beta t-\sigma\omega(t)-\sum_{i=1}^{M_t(\tau)}\log V_i\right\}\right], \qquad (8)\end{aligned}$$

となる [8]．

## 3　モデルパラメータの推定

提案モデルに含まれているパラメータ $\alpha$，$\beta$，$\sigma$，およびジャンプ項に含まれるパラメータ $\boldsymbol{\theta}$ は一般には既知ではないので，実測データなどの利用可能なデータを使って値を推定しなければならない．確率過程 $\Lambda(t)$ の $K$ 次の同時確率分布を

$$P(t_1, y_1; t_2, y_2; \cdots; t_K, y_K) = \Pr[\Lambda(t) \leq y_1, \Lambda(t) \leq y_2, \cdots, \Lambda(t) \leq y_K | \Lambda(t) = 0], \tag{9}$$

とし，その同時確率密度を

$$p(t_1, y_1; t_2, y_2; \cdots; t_K, y_K) = \frac{\partial^K P(t_1, y_1; t_2, y_2; \cdots; t_K, y_K)}{\partial y_1 \partial y_2 \cdots \partial y_K}, \tag{10}$$

とする．$\Lambda(t)$ は連続値をとるので，データ $(t_k\ ,\ y_k)$ に対し，尤度関数を

$$l = p(t_1, y_1; t_2, y_2; \cdots; t_K, y_K), \tag{11}$$

と表す．さらに，対数尤度関数を $L$ とすると，

$$L = \log l, \tag{12}$$

となり，提案モデルでは，未知パラメータ $\alpha$，$\beta$，および $\sigma$ を同時尤度方程式

$$\frac{\partial L}{\partial \alpha} = \frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial \sigma} = 0, \tag{13}$$

の解として得ることができる．

一般的に，確率微分方程式モデルのジャンプ拡散項に含まれるパラメータの推定法としては，モーメント法などが知られているが，異なる確率過程を有するため取り扱いが難しいことが知られている．提案モデルは，Wiener 過程とジャンプ拡散過程の 2 種類の独立した確率過程が混在しているため，ジャンプ項に含まれるパラメータ $\boldsymbol{\theta}$ は，以下の評価関数を最小にするときの値を採用する．

$$\begin{aligned} &\min_{\boldsymbol{\theta}} F_i(\boldsymbol{\theta}), \\ &F_i = \sum_{i=0}^{n} \{\Lambda(t) - y_i\}^2 . \end{aligned} \tag{14}$$

ここで，$\Lambda(t)$ は，運用時刻 $i$ $(i = 1, 2, \cdots, n)$ におけるジャンプ拡散モデルにより推定された総開発工数であり，$y_i$ は実際の累積開発工数を表す．また，$\boldsymbol{\theta}$ はジャンプ項に含まれるパラメータを表す．本論文では，遺伝的アルゴリズムにより式 (14) の評価関数を最小化する．

## 4　最適メンテナンス時刻の推定

上述したジャンプ拡散過程モデルに基づく開発・保守労力の最適化問題を考える．提案モデルから，OSS の運用に伴う開発・保守労力に基づく最適メンテナンス時刻を推定することが可能となる [9,10]．まず，OSS の開発と運用段階における総開発労力を定式化するため，以下のようなパラメータを定義する．

$r_1$：運用段階における開発工数の重要度，

$r_2$：OSS の運用に必要とされる単位時間当りの運用工数，

$r_3$：メンテナンス後における保守工数の重要度．

このとき，運用段階における開発工数は，以下のように定式化できる．

$$E_1(t) = r_1 \Lambda(t) + r_2 t. \tag{15}$$

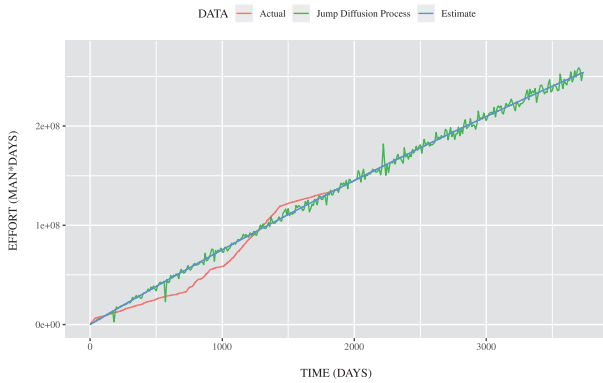


図 1： 指数形ジャンプ拡散過程モデルにおける推定された開発工数.

また，メンテナンス後におけるソフトウェアの保守工数は次式で与えられる.

$$E_2(t) = r_3 \{\alpha - \Lambda(t)\}. \tag{16}$$

上記から，総開発工数は，以下のように定式化できる.

$$E(t) = E_1(t) + E_2(t). \tag{17}$$

式 (17) を最小にする時刻 $t^*$ が最適メンテナンス時刻となる.

## 5 数値例

Apache HTTP Server Project [11] におけるデータを利用した数値例として，推定された投入開発工数として式 (7) および式 (8) におけるサンプルパスを図 1 および図 2 に示す．図 1 から，指数形ジャンプ拡散過程モデルの場合においては，Wiener 過程およびジャンプ拡散過程におけるノイズが大きい様子がわかる．このことから，指数形ジャンプ拡散過程モデルでは投入開発工数が悲観的に見積もられていることが確認できる．一方，図 2 においては，ノイズが小さいことから，安定した稼働が期待できる様子が確認できる．特に，図 1 から，指数形ジャンプ拡散過程モデルの場合において，3000 日目における開発工数は，約 $2.1 \times 10^8$(人・日) となる．一方，図 2 から，S 字形ジャンプ拡散過程モデルにおいては，3000 日目の開発工数は，約 $1.8 \times 10^8$(人・日) であることが分かる．この結果から，将来的に必要とされる開発工数に関して，指数形ジャンプ拡散過程モデルの方が悲観的に推定されていることが確認できる．さらに，指数形および S 字形ジャンプ拡散過程モデルのジャンプ項に含まれるジャンプ幅を表す $V_i$ の推定された密度関数を図 3 および図 4 に示す．図 3 および図 4 から，指数形ジャンプ拡散過程モデルのジャンプの幅が大きくなっている様子が確認できる.

さらに，指数形ジャンプ拡散過程モデルに基づく総開発工数のサンプルパスを図 5 および図 6 に示す．図 5 および図 6 から，指数形ジャンプ拡散過程モデルにおける推定された総開発工数はジャンプ項における影響を受けるが，S 字形ジャンプ拡散過程モデルにおける推定された総開発工数についてはジャンプ項による影響はなく安定していることが分かる.

## 6 おわりに

従来から，ソフトウェアの信頼性や品質を評価するために，ソフトウェア信頼度成長モデルが利用されてきた．ソフトウェア信頼度成長モデルでは，フォールトデータに基づいてソフトウェア信頼性を評

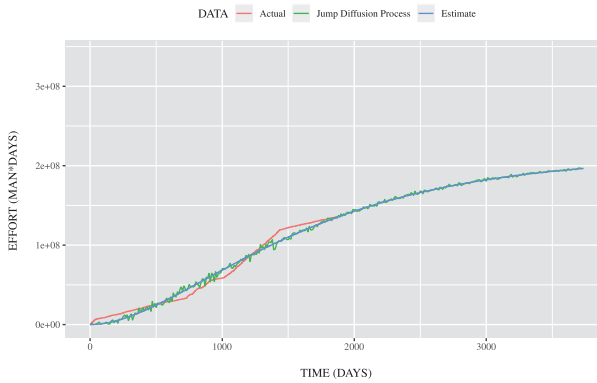


図 2： S 字形ジャンプ拡散過程モデルにおける推定された開発工数.

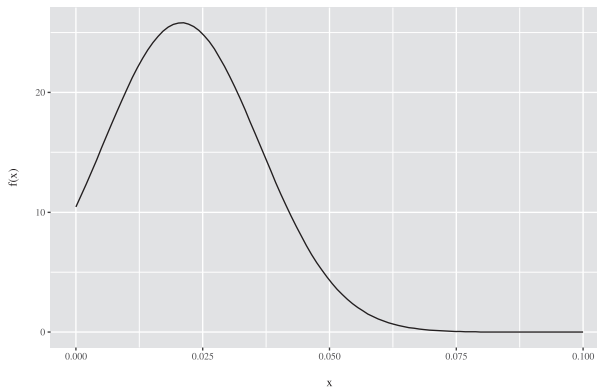


図 3： 指数形ジャンプ拡散過程モデルのジャンプ項に含まれるジャンプ幅を表す $V_i$ の推定された密度関数.

価するが，フォールト発生の原因となるプロジェクト状態を定量的に評価することができれば，ソフトウェア開発と運用において QCD（Quality，Cost，Delivery）の観点から最適化を図ることが可能となるものと考える.

本論文では，OSS 開発プロジェクトの運用段階において，開発工数を予測するためのジャンプ拡散過程モデルを提案した．さらに，提案されたジャンプ拡散過程モデルに基づく開発・保守労力の最適化問題として，OSS の運用に伴う開発・保守労力に基づく最適メンテナンス時刻の推定手法を提案した．また，バグトラッキングシステム上に登録されているデータを利用した提案手法に基づく数値例を示した．過去に，フォールトデータに基づく確率微分方程式モデルやジャンプ拡散過程モデルが提案されてきたが [8]，フォールトデータと比較して開発工数の方が実測および推定結果の値が非常に大きくなることから，ノイズを考慮することを考えた場合においてモデルの振る舞いにおける具体性が高くなることから，より現実的な評価が行えるものと考える.

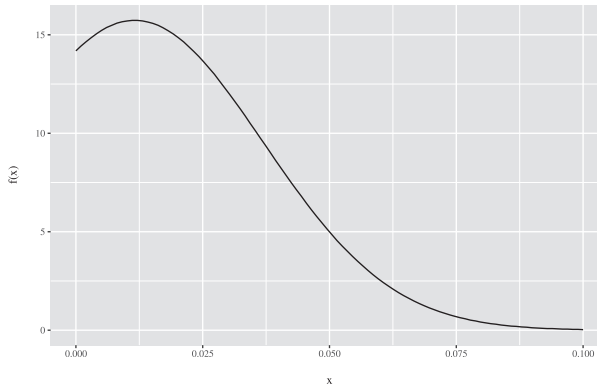


図 4： S 字形ジャンプ拡散過程モデルのジャンプ項に含まれるジャンプ幅を表す $V_i$ の推定された密度関数.

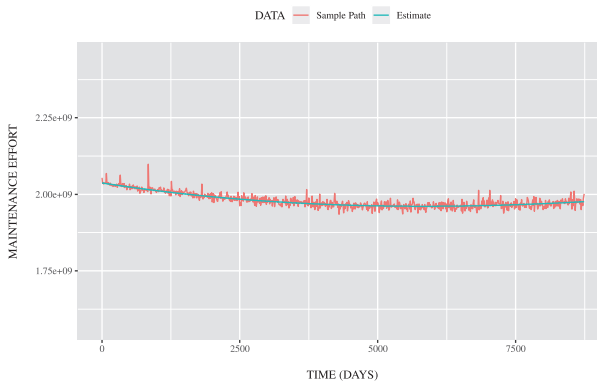


図 5： 指数形ジャンプ拡散過程モデルにおける推定された総開発工数.

## 謝辞

本論文の一部は，JSPS 科研費基盤研究（C）（課題番号 16K01242）の援助を受けたことを付記する.

## 参考文献

[1] I.M. Ibrahim et al., “A robust generic multi-authority attributes management system for cloud storage services,” *IEEE Trans. Cloud Computing*, 10.1109/TCC.2018.2867871, 30 Aug. 2018.

[2] S. Yamada, *Software Reliability Modeling: Fundamentals and Applications*, Springer–Verlag, Tokyo/Heidelberg, 2014.

[3] P.K. Kapur, H. Pham, A. Gupta, and P.C. Jha, *Software Reliability Assessment with OR Applications*, Springer–Verlag, London, 2011.

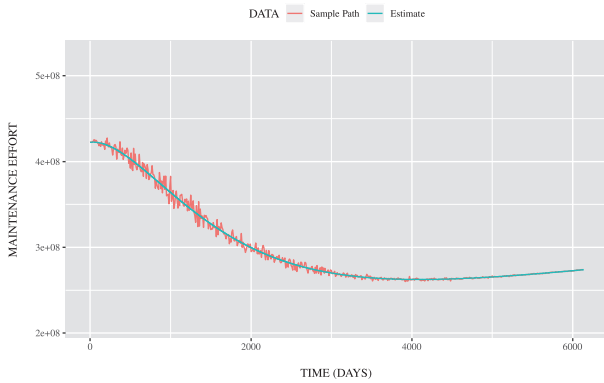


図 6： S 字形ジャンプ拡散過程モデルにおける推定された総開発工数.

[4] L. Arnold, *Stochastic Differential Equations–Theory and Applications*. John Wiley & Sons: New York, 1974.

[5] R.C. Merton, "Option pricing when underlying stock returns are discontinuous," *Journal of Financial Economics*, Vol. 3, Issues 1–2, pp. 125–144, 1976.

[6] E. Wong, *Stochastic Processes in Information and Systems*. McGraw–Hill: New York, 1971.

[7] S. Yamada, M. Kimura, H. Tanaka, and S. Osaki, "Software reliability measurement and assessment with stochastic differential equations," *IEICE Transactions on Fundamentals*, Vol. E77–A, No. 1, pp. 109–116, 1994.

[8] Y. Tamura, T. Takeuchi, and S. Yamada, "Software reliability and cost analysis considering service user for cloud with big data," International Journal of Reliability, Quality and Safety Engineering, Vol. 24, No. 1, World Scientific, pp. 1750009-1–1750009-14, 2017.

[9] S. Yamada and S. Osaki, "Cost-reliability optimal software release policies for software systems," *IEEE Transactions on Reliability*, Vol. R–34, No. 5, pp. 422–424, 1985.

[10] S. Yamada and S. Osaki, "Optimal software release policies with simultaneous cost and reliability requirements," *European Journal of Operational Research*, Vol. 31, Nol. 1, pp. 46–51, 1987.

[11] The Apache Software Foundation, The Apache HTTP Server Project, http://httpd.apache.org/